Detection and Tracking by Wireless UWB and Camera Networks

Toni Axelsson, Ludwig Brandt and Christian Olsson
School of Electrical Engineering
Royal Institute of Technology, KTH
Stockholm, Sweden

***Abstract*—In this report, we have focused on how wireless Ultra Wide-Band (UWB) and camera networks can be used to detect and track moving objects. The aim of the project is to investigate how these technologies can be integrated into the smart building, where there are a vast number of applications that can benefit from accurate position information.**

This report describes the theory behind wireless UWB and camera networks, and image processing algorithms for detection and tracking using vision information. The difference between 2D and 3D localization has been investigated and a method to perform 3D localization using only two cameras is proposed. Localization by multiple cameras can be refined using an Extended Kalman Filter, and this report describes how this is done. Using MATLAB and a webcam, we have implemented different background subtraction algorithms, and then used them for tracking moving objects. In collaboration with project D4, a method to fuse the information from a wireless camera network with an AGV is discussed.

Finally, a new application using a wireless camera network called "The Smart Illumination System" is proposed. This application can be used to optimize the lighting in the smart building, and decrease the energy consumption and at the same time, make the everyday life more comfortable.

***Index Terms*—Wireless Camera Network, UWB, Image Processing, Background Subtraction, Kalman Filter**

I. Introduction

A. Background

INFORMATION technology and electronic devices are today to a great extent incorporated in our homes and buildings. The problem however is that many of the appliances are working independently and were all fitted after the buildings were constructed. In an effort to save time and energy, the idea is, with the help of network controlled infrastructure, to get the devices to work together in an efficient and intelligent way. This would basically mean that the integrated electronics in the building would work more efficiently by sharing information, and become automatically or semi-automatically controlled. Examples of such devices and controllers that would benefit from smart integration are: climate control, surveillance, and domestic robots amongst others.

One of the most important technical problems is to be able to detect and track moving objects in indoor environments, which is what we have focused on in this project. This would result in a vast number of applications which could be applied in the smart building. For instance, in climate control, the ability to detect the number of people in a room could be used to adjust the heating and ventilation for optimal effect and decreased energy consumption. In home security, tracking systems would be a vital part in detecting moving objects and identifying possible intruders. For controlling domestic robots, it is a high priority to know the position of the robot and detect any possible obstructing objects.

B. Problem Formulation

Nowadays, there are several methods and technologies available for tracking and detecting moving objects. In this project we have focused on using low power wireless sensor network (WSN) camera systems and Ultra Wide-Band (UWB) radio sensors. We have studied these technologies and found efficient ways of using them to detect and track moving objects. We have also studied how the information available from different sources can be fused to make the system more precise and reliable. There are several problems and difficulties that should be considered before implementation, such as: video resolution, system accuracy, and costs. We have implemented algorithms for detection and tracking that can be used in a WSN camera system, using a simple webcam and a computer. After studied and testing these algorithms, we have proposed methods on how to use them and ways to solve certain problems. Finally, with our acquired knowledge, we have suggested a new application which we can benefit from in smart buildings.

The rest of the report is organized as follows. In Sections II and III, we present the theory behind the technologies. We follow this section by our implementations results in Section IV. In Section V, we propose methods for localization and information fusion. Our new application is presented and discussed in Section VI. Finally, we conclude the report with discussions and suggestions on future work in Section VII.

II. Theory of WSN Camera Systems

A. Wireless Sensor Networks (WSN)

According to Wikipedia a WSN "consists of spatially distributed autonomous sensors called sensor nodes, with the ability to cooperatively pass their data through the network to a main location" [1]. Another definition of a WSN is a large number of low cost and energy efficient nodes with sensing, computing, and wireless communication capabilities [2].

A typical sensor node consists of five different components. The first component is the sensor, which has the ability to measure different conditions in the environment. Examples of sensors that could be used in a WSN are microphones, thermometers, pressure sensors, or image sensors [3]. The choice of sensor depends on the desired application of the WSN [2]. A microcontroller that processes the data from the sensor and a memory are also components of the sensor node. Another component is a wireless communication device, to transmit (receive) information to (from) other sensor nodes. The last component is the energy source, e.g., a battery [3]. Depending on the application the number of sensor nodes deployed in a WSN could vary from a couple to several hundreds [1].

The purpose of a sensor network is that each sensor node separately collects data and transmits the information to a base node, and then often to a central computer, where all the data is fused [4]. WSN has a wide range of applications, such as home automation, healthcare monitoring, and military applications [1].

B. Visual Sensor Network

An example of a wireless sensor network is a visual sensor network, or wireless camera network, where the sensor nodes are equipped with cameras [2]. The idea behind the visual sensor network is to process and fuse images from a variety of viewpoints, and therefore, extract more useful information from them rather than from a single camera. [5].

The difficulty with camera nodes is the amount of information that the image sensor provides, compared with sensors such as thermometers or microphones. Most sensors provide information as 1D data signals, while image sensors provide 2D data signals that form an image. As a result, the complexity of processing the information, and bandwidth needed for communication increases [6]. For many applications, such as detecting and tracking moving objects, the visual sensor network must be able to process and transmit information from the camera nodes in real-time [7]. Also costs and energy consumption has to be considered depending on the application. If a visual sensor network is applied in "the smart building" the cost and energy consumption cannot be very high.

Examples of visual sensor nodes that are available today are MeshEye, Cyclops, and CMUCam3. As an example of hardware in a sensor node, the CMUCam3 has a microcontroller with clock rate of 60 MHz, image sensor with resolution of 352×288 pixels, and the communication device has a transfer rate of 250 kbit/s [6].

C. Image Processing in Visual Sensor Networks

Image processing in a visual sensor network could be performed in different ways. One solution is to send the gathered images from the camera nodes directly through the network and then perform processing at a central computer. The drawback with this method is the amount of information that has to be sent, which put demands on the bandwidth of the network [2].

Another method could be to process the images at camera nodes locally, using the computational capabilities of the node. By doing this, the total amount of data that has to be sent through the network decreases, because the camera node could choose what information to transmit [6]. If the camera nodes could perform simple image processing algorithms, only the most important information has to be communicated. An example of an image processing algorithm that could be performed at the camera node is object detection [2]. If no moving objects are detected there are no reason to transmit the captured images. The complexity of image processing algorithms that can be performed at the camera node depends on the capacity of the microcontroller and available energy. The selected information from the camera nodes is then fused together and could be used in more complex processing algorithms [6].

The techniques used for image processing comes from the vision computing field. It is important to remember that many vision computing algorithms has to be customized for implementation in a visual sensor network. The reason is that the algorithms may require much more powerful processors and memory resources than what is available at the sensor nodes [8].

D. Object Detection Using Vision Information

As already mentioned, detection of moving objects is a technique to process the captured images at the sensor nodes, used in order to reduce the amount of information that has to be transmitted through the network. There are many different algorithms on how to perform object detection using vision information from the camera. The common approach is to make a difference between background and foreground. In that way, irrelevant stationary objects are ignored since they become parts of the background and moving objects become parts of the foreground. Three techniques on how to separate background from foreground are: offline background subtraction, frame by frame subtraction, and adaptive background estimation [9].

1) Offline Background Subtraction: When performing an offline background subtraction, one starts with taking an image when no moving objects are in the view and save that image as the background. To find moving objects, one compares every new image with the saved background image, by subtracting the background from the new input image pixel by pixel. [9].

A mathematical description of offline background subtraction is

$$|I_n(x, y) - B_0(x, y)| > T.$$

$I_n(x, t)$ is the color intensity at pixel (x, y), in the input image frame number n. $B_0(x, t)$ is the color intensity at pixel (x, y), in the background image and T is a threshold. Now, if the absolute value of the subtraction is larger then T, the pixel (x, y) will be detected as moving and will become a part of the foreground. If less than T the pixel will stay as a part of the background. The threshold value T is used because of the risk that the background could slightly vary from image to image and that should not be detected as motion [10].

One drawback of offline background subtraction is that it does not take into account that the background can change. This is especially evident if the camera is placed outside or near a window, where brightness, time of the day and weather condition will change the background [9].

2) Frame-by-Frame Subtraction: To handle the problem with the changing background, we can use a technique called frame-by-frame subtraction. Instead of having a static background image for comparison, the solution is to continuously comparing images to detect motion. This is done by subtracting the input image with the last or second to last input image. Certainly, the image will always change gradually frame by frame creating a noise. Using a given threshold when comparing pixels, small changes are ignored and are not detected as movement. The mathematical description is

$$|I_n(x, y) - I_{n-1}(x, y)| > T.$$

Contrary to the static background subtraction, the background would constantly change to the previous video frame. As in offline background subtraction if a pixel is different from the previous image it will become a part of the foreground [11]. One downside is that if a moving object comes into the view the algorithm will detect it, however, if the object then becomes stationary, it will instantly become a part of the background [9].

3) Adaptive Background Subtraction: One approach to handle stationary objects is to use an adaptive background model. In such model, we can incorporate different changes such as brightness and other scenery changes at the price of not being able to detect objects that have been stationary for long time [9]. The adaptive background subtraction uses a running statistical average of the intensity at each pixel. If the value of the intensity at a pixel is significantly different from the average intensity, the pixel will be marked as a possible moving pixel [12]. The two most general approaches in adaptive background estimation are parametric estimation and non-parametric estimation. The difference between those two is that the parametric estimator assumes that the background is distributed in a predefined way, while in the non-parametric case, the background can be distributed arbitrarily [9].

E. Object Tracking Using Vision Information

Once an object has been detected, the next step is to track the movements of the object. There are many available approaches for tracking objects. The first issue to solve is how to represent the moving object. The most common and intuitive ways representing objects are points, color, shape, contour, or motion [9]. The choice of representation does influence the complexity of the image processing algorithms that can be performed when the vision information from the sensor nodes is fused. For example, it would be impossible to perform object or behavioral recognition, if the object is represented as a single point at the sensor nodes.

The next step is to select the features of the moving object. Features of an object are characteristics that make the object distinguishable from the background and easier to track. The feature selection is related to the choice of representation [13].

1) Region Based Tracking: A region based tracking algorithm tracks objects using variations of image regions between image frames and matching them with moving objects. These moving image regions, called blobs, are generated through background subtraction [14]. These moving regions can be both represented as a single point in the centre of the blob or as several points. The blob can also be represented as a shape.

When detecting an object, an easy feature to distinguish is color. However, the object should have a clear and distinct color for the tracking to be efficient. For instance, in the case of tracking human targets in a building, it is very unlikely that they all walk around with one specific color on their shirts, unless you are in a place where people carries uniform. This is not an issue when tracking targets, such as a robot since we could paint areas with specific colors [9].

2) Contour Based Tracking: Contour based tracking is an algorithm that extracts the contours of the moving objects and uses them as a representation. An advantage of this algorithm in comparison to region-based tracking is that contour based tracking describes objects more simply, which reduces computational complexity. Another advantage is that contour based tracking can track objects even if the object is partly occluded, e.g., if a person standing behind a couch [14].

3) Shape Recognition: For detecting humans or other objects with known shape, we could use shape recognition and therefore, prevent figures that look like humans from becoming a part of the background when being stationary. However, the human body has a very complicated shape, which leads to problems developing efficient algorithms for recognizing the human body. One solution is to concentrate on special parts of the human body that are easier to recognize. For instance, the legs move much more than the upper body, etc.

4) Optical Flow: Another way of detecting and tracking moving objects, that does not involve background subtraction is a method called optical flow. The idea behind optical flow is that all the moving points in the image has a flow vector, describing the direction and magnitude of the motion. The flow vector is calculated by observing the position of a given point in two consecutive image frames. In the mathematical description of optical flow we make the following assumption

$$I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t).$$

The assumption says that the pixel (x,y) has moved a distance $(\Delta x, \Delta y)$, between two consecutive image frames. Δt denotes the time between the two image frames. The next step is to assume that the distance is small, i.e., the moving object has not moved far during the time Δt. As a result, $I(x,y)$ can be expanded with a Taylor series. Neglecting the higher order terms results in

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t.$$

Then, by using (**??**), we get

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0.$$

Dividing both sides by Δt gives

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0.$$

This can be written as

$$I_x V_x + I_y V_y = -I_t,$$

where V_x and V_y is the optical flow of $I(x,y,t)$. The problem is that this equation cannot be solved for the two variables V_x and V_y. Therefore additional assumptions have to be introduced to calculate the optical flow .

Lucas-Kanade method is one approach for calculating the optical flow. This method assumes that the flow is constant in small regions around each pixel. The method then uses the optical flow equations and the least square method in that region, to find the optical flow [15]. Methods for calculating the optical flow has a tendency to be computationally complex, which could be challenging in a visual sensor network [11].

5) Filter for noise reduction: There are several situations where noise can influence the object detection and tracking. The noise in image acquisition can be seen as single or a small groups of pixels displayed in a deviant color compared to their neighboring pixels. Algorithms such as the offline background subtraction rely on a static background and a clear foreground when comparing images. If the background has changed slightly due to, e.g, lighting, these parts will be interpreted as foreground. However, by applying a smart filter, the algorithm can easily eliminate small noises in the images without compromising the object detection or tracking. Shadows cast by the moving object are also a problem for the algorithms since they cause light changes in greater areas. These shadows are often much larger than a simple noise and a standard filter might not be able to remove their effect. In these cases, one should employ a separate shadow removal algorithm [11].

III. Theory of UWB Radar Sensor Networks

A. Ultra-Wideband Technology

The Ultra-wideband (UWB) is a high-bandwidth radio technology, which can be used at very low energy levels for short-range communications by using a large portion of the radio spectrum (frequencies lower than 300GHz). The UWB was traditionally said to be a pulse radio, but today one defines it as "A transmission from an antenna for which the emitted signal bandwidth exceeds the lesser of 500MHz or 20% of the center frequency". Modern applications of the technology are seen in various UWB sensors and radars, which have proven useful in sensor data collection, object detection, and tracking. In order to optimize the sensor data, one would connect several sensors and/or radars together with "a system base node" in what is called a "UWB sensor network" [16].

B. The UWB Sensors

In object tracking and detection applications, short-range UWB sensor networks can be used in a typical indoor environment. This is partly achieved by a technique called "UWB imaging", which utilizes radar transmitters (Tx) and receiver antennas (Rx). The radar that is mostly used for UWB imaging is a synthetic-aperture radar (SAR), characterized by using the relative motion between a moving antenna and a stationary transmitter. The Tx emits a periodical short electromagnetic pulse in a certain time interval; these pulses are then scattered at nearby objects and later arrive at the Rx. The delay between emitted and received pulse is proportional to the distance the pulse has traveled. In order to produce an image, readable for people, one needs to merge the pulse information using *migration algorithms*. In cases where a mobile Rx is not possible due to lack of free space, multiple stationary sensor nodes with transmitting and/or receiving properties can be used instead. In such network configurations, objects can be localized by a combination of range measurements from several sensor nodes [17].

C. Sensor Network Nodes

The construction of a reliable and robust indoor sensor network depends on the usage of different types of nodes and clever positioning of them. The basic nodes have either transmitting or receiving antennas, while the more sensitive nodes are capable of having one receiving and one transmitting, or even two receiving antennas in the same node. The node with two Rx antennas, called "Scouts", are very useful for detecting and recognizing the features of the environment on their own. Multiple scouts are commonly used for extracting partial maps of their surroundings and identifying unknown objects in a detailed way. The key nodes in the sensor network are the so-called "Anchor Nodes", which have both Rx and Tx capabilities. The location of a deployed anchor node is very important since it serves as a the main transmitting nodes which must be able to transmit to all Rx-only nodes, thus spanning a large volume of the selected environment.

In order to work optimally, all sensor network nodes need to be synchronized, sometimes only temporary, in order to cooperate and perform object and environment detection. In a new unknown environment, every sensor needs to know their relative position in the environment by deploying their own local coordinate system. Secondly, the nodes needs to know where all the other distributed senors are located relative to themselves. Lastly, the structure of the unknown environment is to be recognized using imaging and object recognition methods. When the surroundings of the node has been identified, the node will relate the information to its own coordinate system. The quality of the imaging process depends considerably on the positioning of the deployed nodes. If the anchor nodes are able to identify the surrounding environment, while Rx nodes being available simultaneously, then the information between nodes can flow smoothly. This makes it possible for real-time processing which is greatly used in the fields of object tracking [18].

D. UWB versus Continuous Wave Radars

The previous technique is mostly beneficial in search and rescue environments, e.g. burning buildings or other catastrophes, although the general technique for object detection and tracking can be applied in other scenarios and suited for more daily situations e.g. climate control [19]. The UWB radar has some key advantages compared to continuous wave radars:

1) The wide frequency pulse can easily pass through obstacles;
2) The pulse duration is very small, which allows for a high resolution;
3) The short pulse leads to low energy consumption;
4) The UWB radar is able to give an exact position of a detected object.

E. UWB Imaging of Unknown Environments

In order to create an image of a static environment and detect crucial objects, the usage of a stationary anchor node (or several nodes, depending on environment size etc) and one observer node (double Rx) is sufficient. While the anchor node is transmitting the radar pulse, the observer node is moving along either a predesignated or arbitrary path, collecting data from the backscattered waves. The natural method for UWB imaging is Time-of-Arrival (ToA) based localization. While the observer is moving along the path, the image is gradually being built and enhanced. In order to produce a valid image, the pulse velocity is predetermined and the distance between anchor node and receiver node is always being computed.The basic principle of migration imaging can be seen in Figure 1. A transmitted signal is sent from the Tx at $[x_t, y_t]$, which later scatters at the object at $[x_0, y_0]$. The signal eventually ends up with a time delay τ at the Rx at variable $[x_{Ri}, y_{Ri}]$. Assuming a single reflection, the received signal and the recorded time delay will create an elliptical location of

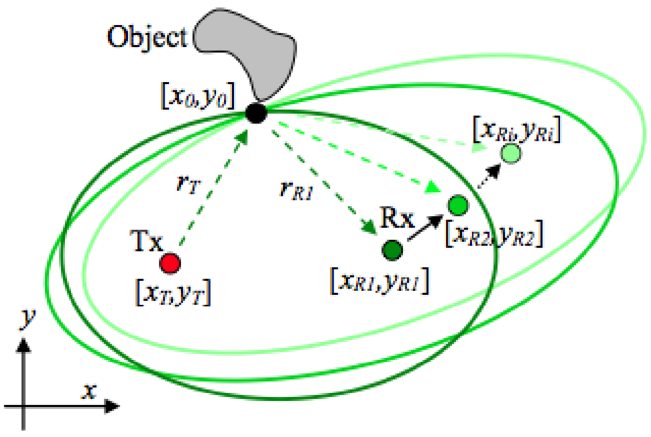

Fig. 1. Basic principle of migration imaging [18]

where the object might be. However, the more the observer moves, the ellipse will change its appearance apart from one point. This common point is $[x_0, y_0]$, and it will be the point that is focused in the image at the correct pixel position [18].

F. Object Detection Using Stationary UWB Radar Sensors

The previously explained technique requires a moving receiver node, preferably located on an autonomous vehicle or set up at a stationary track. The major advantage with the moving Rx is that the total number of deployed sensor nodes can be kept down, and resulting in slightly more energy efficient and less complex implementations. In cases where a moving Rx node is not preferred, e.g. in a civil building, several Tx and Rx nodes can be deployed in total collaboration and still perform accurate tracking and detection. In such networks, multiple Tx and Rx are needed in the environment, occasionally together with a radar sink (depending on the size of the environment). The sink is used in scenarios where one has a very large area to cover, and many sensor nodes. In order to save energy, only a fraction of the nodes are always running. As soon as an object is detected, the discovering node alerts the sink which responds by activating the nearby sensors for optimal object detection. In smaller spaces, such as offices or domestic house rooms, we have less nodes and hence the role of the sink becomes less important. To detect an object and plot its position, the elliptical localization between two nodes is compared with other nodes in the network to ultimately find the detected object's position in the network [20].

G. 3D Imaging

In order to perform a 3D imaging and positioning, one would often use several 2D measurements and combine them into a 3D image. Each measurement would extract a 2D image of the object with the information on how far away the object is and where in the plane the object and its contour lie. The first used measurement would be the plane which lies closest to the sensor, where the contours are extracted and saved. The next measurement will be a parallel plane which lies a little further back compared to the first plane. The contours from this plane are now saved and combined with the first plane's contours into a 2D image. The following measurements would continue to step through parallel planes and record the contours until the whole object has been covered. The end product is then a 2D image with different contours at specific levels, and from this image it is now possible to reconstruct a 3D image of the object using a computer.

To perform 3D imaging this way, one UWB radar will only 3D map on side of the object, leaving "the back" of the object undiscovered. A second radar sensor placed on the other side of the object could then in collaboration create a complete 3D image of the object covered from two angles. However, if one is interested in measuring the size and/or volume of an object it would be most simple to use three UWB radar sensors and make sure they cover all necessary angles and fuse the measurement data to create a complete 3D image of the detected object. The number of required UWB radars to perform 3D imaging is ambiguous, and depending on the detail of the 3D map the number of sensors vary. One sensor might be enough, but for a more complete 3D image multiple UWB radar sensors are needed [21].

IV. Implementation of Background Subtraction Algorithms

To gain a better understanding of image processing algorithms that could be performed at the senor nodes in the visual sensor network, we have implemented two algorithms for detection of moving objects. The methods that we have focused on is offline background subtraction and frame-by-frame subtraction. The reason for choosing these algorithms is that the theory behind them, is rather simple, and we already have the necessary equipment and programming skills to implement it. A requirement which has to be taken into account is that the algorithm must be suitable for the sensor network node, i.e. low power consumption. In our case, our computer was much stronger than the average sensor node, but the recording device was set to a very high resolution which slowed down the algorithm.

For the implementation, we used MATLAB environment and a built-in webcam on an iMac. The webcam's settings were limited to a resolution of 1280×1024 pixels, and color scheme YCbCr, which had to be converted to RGB colors in order for MATLAB to work with it. Each capture frame is saved as three dimensional $n \times m \times 3$-matrix where n = *height in pixels* , m = *weight in pixels*, and *3 = RGB-value*. The 3 RGB-values in the third dimension of the matrix each represent the three colors, 1 being red, 2 green and 3 blue, e.g. the value at $(500, 200, 1)$ represents the red color level at pixel $(500, 200)$, and $(500, 200, 2)$, and $(500, 200, 3)$ represent the level of colors green and blue in that pixel. These values vary between 0 - 255, where 0 is total lack of the color, and 255 is maximum amount. We use the following notation to show the image

$$\mathbf{I} = \begin{bmatrix} p(1,1,RGB) & \cdots & p(1,1280,RGB) \\ \vdots & \ddots & \vdots \\ p(1024,1,RGB) & \cdots & p(1024,1280,RGB) \end{bmatrix}.$$

A. Offline Background Subtraction

In our implementation, the camera records while MATLAB saves the frames in matrix format. The captured images are saved in a sequence $I_0, I_1, ..., I_{n-1}, I_n$, where n is the total number of frames. In background subtraction, the first captured image I_0 serves as a reference for the algorithm. It is therefore important that the reference image is solely made of the static background. Before the images are compared, the captured color image needs to be converted into a grayscale image. Mathematically this means removing the third dimension of the picture matrix $\mathbf{I}$. This has been done using a built-in MATLAB function. The matrix is still of size 1280×1024, and each matrix entry has a value between 0 - 255, where 0 being black, 255 white and everything in between is a shade of gray.

As soon as the second image is captured, the algorithm investigates each pixel individually through a *for-loop* and then compares it with the same pixel in the reference image. If the pixels vary more than a certain T = *threshold value*, the pixel is said to belong to the foreground. If the variation is less than T, then it is considered as background.

The equation (1) is the calculation that is performed at each pixel, P, in the image $\mathbf{I}$. The subscript n denotes the number of the input frame in the sequence, and the subscript 0 is the background frame.

$$|P_n - P_0| \geq T \tag{1}$$

After each pixel comparison, the algorithm builds up a new image pixel by pixel. This new image is only black and white, where pixels belonging to the background being black, and the foreground being white. Practically, this means that if a pixel was identified as background, its pixel value is set to 0, and if it is part of the foreground the pixel value is set to 255. The new images are kept in a sequence which can later be displayed as a video. In Figure 2, we demonstrate the algorithm separating the background from the moving object in frames 1-4. The two following frames show our static background with and without a moving object in original RGB-format.

B. Frame-by-Frame Subtraction

When implementing frame-by-frame subtraction, the reference image that the algorithm compares the input image with, will be changed to the previous image in the sequence at every comparison. The threshold condition will in this case be

$$|P_n - P_{n-1}| \geq T.$$

Figure 3 shows that this algorithm does not detect the moving object as clear, compared with the offline background subtraction algorithm. However, this algorithm

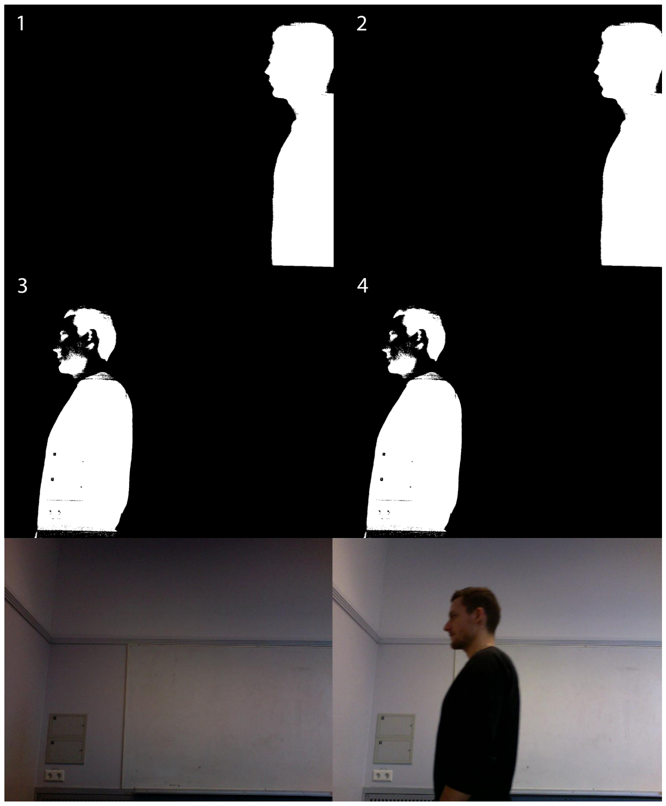

Fig. 2. Results of our offline background subtraction algorithm, together with the static background and object in color

handled changes in the background. One setback with this algorithm is that if the object does not move much between frames, the interior of the object will be seen as black. If the object does not move at all it will disappear, which was evident during our experiment.

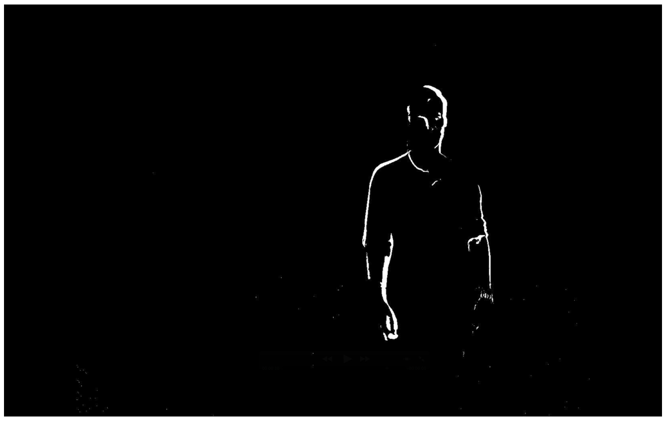

Fig. 3. Results of our frame-by-frame subtraction algorithm

C. Noise Reduction

A background subtraction algorithm will work best, if there is a strong distinction between background and foreground with clear black and white images. However, this is far from reality since the background subtraction algorithm investigates each pixel individually for color changes. If the brightness of a room changes during the process, color of the background would either become brighter or darker, resulting in a change in pixel values. If the brightness variation is large enough in a pixel, the algorithm would interpret it as a movement and hence color it white in the product image. This is called *noise* and can be seen as random white spots in the image which need to be cleared in order to be able to track the actual movement. In our MATLAB-implementation, we created a crude and simple filter with the key ability of being able to sort out small disturbances without using much computational power, and hence, being very quick. The basic principle was to divide the input image into an even number of equally sized sectors, scaled down by dividing the original height and width with 128. When using a 1280×1024 pixel image, we divided the image into sectors of 8×10 pixels. The algorithm determines how many percentages of the sector is white. Now, if more than 20% of the pixels are white, then the sector is in the foreground and left untouched. However, if less than 20%, the sector is considered to only contain noise and belong to the background. The filter then repaints the white pixels black. After successfully filtering a sector, the algorithm continues with the adjacent sectors until the whole image has been filtered. The threshold value in

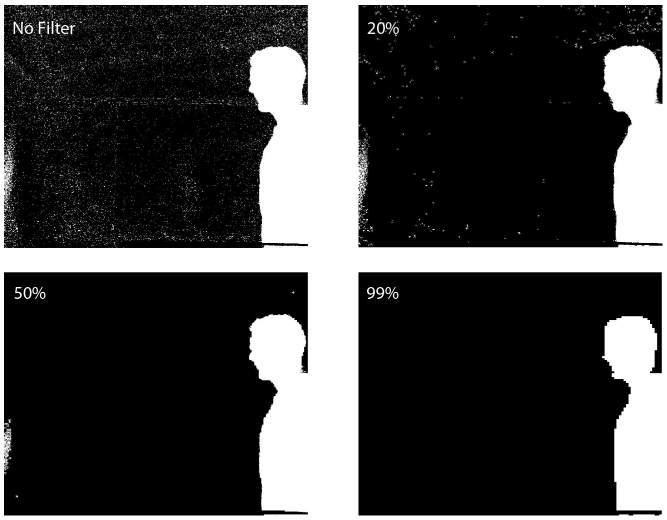

Fig. 4. Results of our filter for noise reduction

percent is of major importance in the filter, and changing the value alters the picture significantly. In Figure 4, we see that small noise elements can easily be filtered out using a 20% threshold, while larger disturbances remain. Increasing the threshold to 50% causes the filter to remove both small noises but also larger ones. However, with increasing threshold value, the actual object's contour (the depicted person in Figure 4) is also being filtered out, creating a more edgy contour. If the threshold is increased all the way to 99%, all noise is successfully removed to the cost of the object's contour which is greatly compromised It is clear that the threshold value has to be chosen so that enough noise is filtered but without compromising the detail of the object.

D. *Single Object Detection and Tracking*

After successfully implementing offline background subtraction and a basic filter, we are now ready for the final implementation of an object detection and tracking algorithm. This program first utilizes our offline background subtraction in order to make the images in the series into binary. We could have moved on without the filter, but since our tracking algorithm scans the image for white pixels when identifying the object, shadows and noise would be apparent, and the algorithm could mistakenly identify a shadow as a real object. Hence, we applied a 99% filter for enhanced performance.

The algorithm itself is very simple and can detect an object and find its position at the cost of handling only one object. It starts to search from the image's upper left corner pixel and continues to the right until it reaches the images width. When the algorithm finds a white pixel, it assigns that pixel the labels: top, left, right and bottom of the object. Assuming that the first pixel found represents the object's top left corner, the algorithm continues to scan for pixels, and for every pixel found further right or down of the top left corner pixel, the labels *right* and *bottom* changes. By doing this, we are able to find four corners from which we then can draw a rectangle in the image, which contains the object. In the case that the algorithm finds a pixel further left of the first pixel, it will change the pixel position as label *left.* The position of the object is then calculated as the center of the rectangle, or the mean value of the *top, bottom, left & right* labels. In Figure 5,

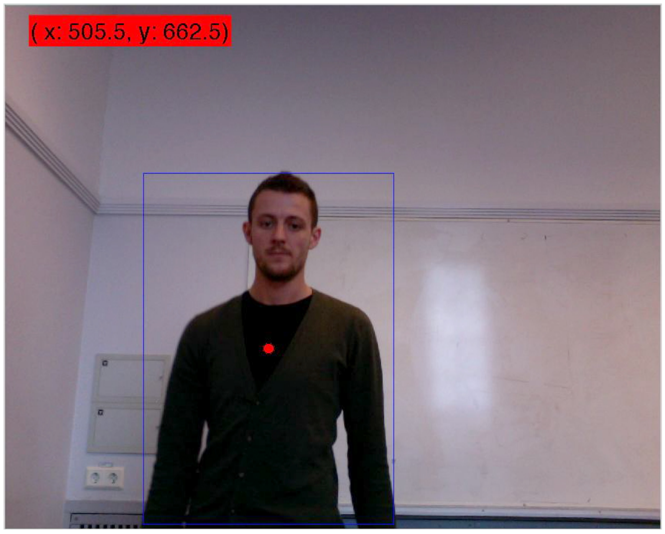

Fig. 5.

you can see an example of this method, the red dot is the center of the rectangle and the object's center pixel position is shown in the upper left corner. This algorithm can successfully identify one object and continue to track it while remaining in the camera's sight. The downside of this algorithm is that if a second object were to enter the image, the algorithm would not be able to tell them apart and instead detect both objects as one big single one. The upside with the algorithm is its simplicity and the fact that it does not need very much computational power.

E. *Multiple Object Detection and Tracking*

When implementing the tracking algorithm, handling multiple objects is a much more complex problem. We focused on handling multiple objects without classifying the objects, they are just labeled "Object" instead of having an identity. In our algorithm, we tried to use distance to separate the objects. If it is a space, in this case black pixels, between the white pixels of a certain number the algorithm makes a rectangle around the white pixels it have found and then begin a new search in the same frame for more objects. The space can be in both x or y directions. You can see the result of implementing this algorithm in Figure 5. However, we noticed that it is much easier to separate objects in the horizontal direction because the way that our algorithm searches through the pixels is that it goes from left to right and then down, like when reading a book. In the vertical direction, it is more complicated. If you imagine having two objects and one is at the upper right corner and one at the lower left corner, and they are overlapping each other in the vertical direction. The algorithm will then find the when one to the upper right first and start to build a rectangle. However, when it comes down to the left one it still have not finished the right object but it needs to start apply values to the left one. Therefore, it needs a way to handle these situations and keep the data it have found separated. The downside with using a specified separation distance is that if the objects get close to each other they become one object. You cannot set the distance very low because you risk splitting one big object into smaller ones.

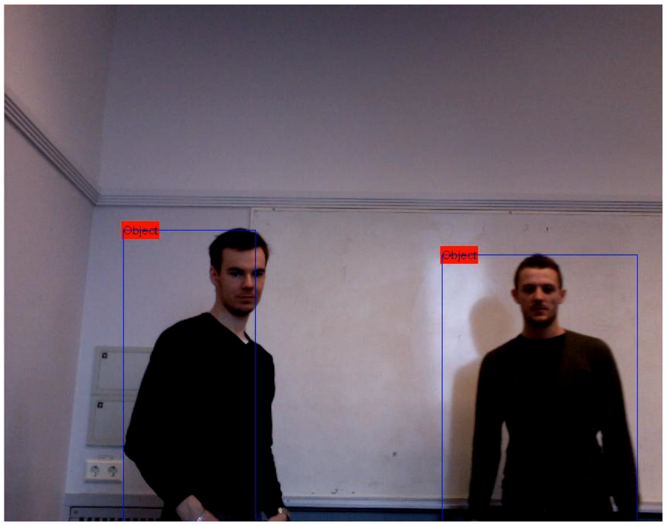

Fig. 6.

V. Localization and Information Fusion

The purpose of localization is to determine the position of a moving object. To be able to localize a moving object, obviously the object first has to be detected and then tracked, as we have described in earlier sections. There are three different aspects that has to be considered for an accurate localization. The first aspect is camera calibration, which is to define the locations of the camera nodes relative to each other. Calibration also contains the orientation of the different camera nodes, that is, the directions that the cameras are pointing. The next aspect is synchronizing their readings to ensure that different cameras are viewing the same scene at the same time. The third step is to perform an estimation of the position based on optics and geometry [22]. The first step for localizing a moving object is to first determine its position in 2D in each individual camera node. The information of the positions from the camera nodes is then fused to find the 3D position of the moving object.

The goal of this section is to describe the differences between localization in 2D and 3D, and to determine the number of camera nodes that is required to perform localization. This will lead to a method for localization in 3D.

A. 2D Localization

The image captured by a camera is a 2D representation of a 3D environment. To find the 2D position of a moving object, we use the pinhole camera model, which is an approximation of the most simple camera without lens. In the model, the rays of light is projected onto a planar screen called image plane, after passing through a pinhole. We start with placing the pinhole at the origin and the image plane perpendicular to the z-axis at a distance $z = f$ from the origin (f is the focal length). We can now calculate how a 3D point $\mathbf{x} = (x, y, z)^T$ is mapped onto the image plane as a 2D point $\mathbf{u} = (u, v)^T$ [10]. Figure 7 shows the geometry of the pinhole model. Looking at Figure 8

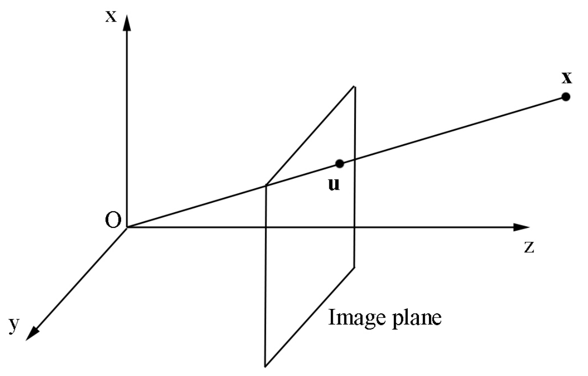

Fig. 7. Pinhole camera model

and using simple trigonometric identities, we get

$$u = f\frac{x}{z}. \tag{2}$$

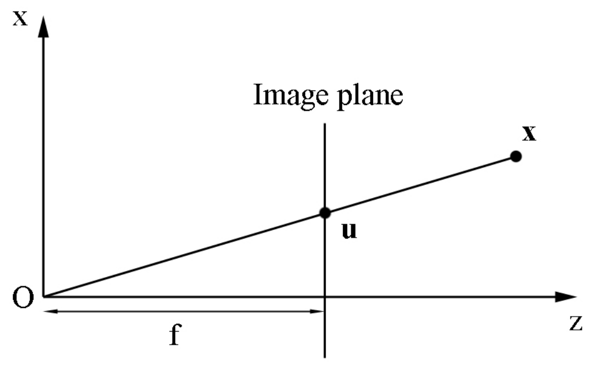

Fig. 8. Pinhole camera model in the xz-plane

In the same way, we can calculate v

$$v = f\frac{y}{z}. \tag{3}$$

Therefore, according to [23], we can calculate the projection of a 3D point onto a 2D image plane as

$$(x, y, z)^T \mapsto \left(f\frac{x}{z}, f\frac{y}{z}\right)^T.$$

The pinhole camera model can be formulated in a more general form, that takes into account the pinhole at an arbitrary position, the image plane orientated arbitrary, and have non square pixels with skew image plane [10]

$$\mathbf{u} = \mathbf{P}\mathbf{x},$$

where $\mathbf{P}$ is called the projection matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}],$$

and $\mathbf{K}$ models the cameras intrinsic parameters, which are focal length, scaling and projection center. The extrinsic parameters are the orientation $\mathbf{R}$ and position $\mathbf{t}$ in the world coordinate system of the camera [24].

Consider a square room, with a camera node located in the center of the ceiling pointing down. The distance from the ceiling to the floor is known. Now, if a moving object is detected by the camera, i.e., a small ball rolling on the floor, it could be represented as a point in the image plan with coordinates $(u, v)^T$. Since the distance z is known, we can use the equations (2) and (3) to obtain the position $(x, y, z)^T$ of the moving object on the floor. This simple example shows that only one camera is required for localization of an moving object in 2D.

B. 3D Localization

We continue our example with the square room and a single camera in the ceiling. If the area of the moving object is known, e.g., the area for a ball being πr^2, then there is a method of calculating the 3D position. If the object gets closer to the camera, it will cover more pixels. If we know the number of pixels that the object is covering at a certain distance from the camera, we can calculate a relationship between the number of occluded pixels and distance from the camera. Then, with the calculated distance z, the 3D position can be found with the same equations as before, using the pinhole model. However, if the area of the moving object is unknown, this method cannot be used.

1) Epipolar Geometry: If we consider two cameras that are viewing a 3D scene, from two different known locations. This setup creates geometric relations between the 3D point and the 2D points on the image planes. These geometric relations are called epipolar geometry, and can be used to determine the 3D position of the moving object [10].

Figure 9 shows the geometry of two cameras with parallel image planes IP_1 and IP_2. Both cameras has focal length f and are located at a distance b from each other. c_1 and c_2 are the horizontal distances between the projections of the 3D point $\mathbf{x}$ on the image planes $\mathbf{u}_1$ and $\mathbf{u}_2$, and the centers of the image planes. The depth z can now be

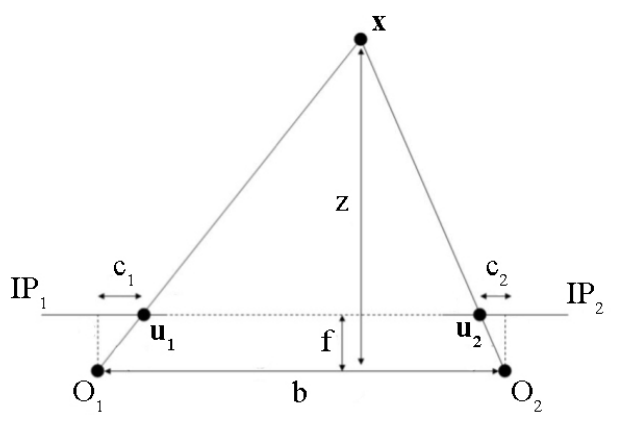

Fig. 9. Geometry of two parallel cameras [23]

calculated using trigonometric identities [23]

$$z = \frac{fb}{c_2 - c_1}.$$

Now, when we have a method to calculate the depth, the 3D position of the object can be determined using the equations from the 2D localization

$$x = z\frac{u}{f},$$

$$y = z\frac{v}{f}.$$

This shows that to localize a moving object in 3D, the minimum number of cameras required are two.

However, there are situations where two cameras are insufficient for determine the 3D position of an moving object. An obvious situation is if the object is occluded and not visible for one or both cameras. The solution to handle this is to deploy more cameras to make sure that at least two cameras are viewing the moving object at the same time.

C. Refined Localization using Kalman Filter

When performing object detection and tracking, it is obviously of high importance to achieve accurate and precise measurements of the position of the object. By using measurement from several camera nodes, one would be able to get a better estimation of the position. The rising problem is, that when multiple cameras are tracking an object, each camera will return a separate estimation of the position of the object. In the optimal situation, all the cameras would return the same position and agreeing on that it is the correct one. In reality, however, this would never occur. This is due to measurement noise, object shape and movement, etc. For example, a object, e.g., a human, have different shapes depending on the viewpoint, which leads to that the returned position will be different, depending on where the camera is located. Other factors creating noise is lens distortion, resolution, and bad background subtraction. Therefore, a method to fuse the position information from multiple cameras would refine the localization. One way of doing this is to, first get an estimation of the 3D location, and then use a Kalman Filter to track the moving object over this estimation [24].

1) Introduction to Kalman Filter: In linear dynamical systems where measurements contain noise, it is difficult for a person to predict what is going to occur in the near future. However, it is of vital importance to be able to predict the next state in a model and cancel out any possible disturbances. To do this, one would use the Kalman Filter which is a recursive data processing algorithm that estimates the state of a linear dynamic system containing noise. With the word state, we mean a vector containing a certain number of variables which describe some property of the system. In object tracking and detection, this vector would usually contain the coordinates and orientation of the object. The methodology and terms used in the Kalman Filter comes from various disciplines in mathematics, such as: least mean squares, probability theory, stochastic system and linear algebra to name a few. Today, the Kalman Filter is being used in modern control theory with applications in aircrafts, manufacturing processes and robots among others [25].

The Kalman Filter estimates the state of the system by using all the measurements of the system. Those measurements need to be linearly related to the state and they are often more or less corrupted by various noise. The process then the available measurements, both accurate and inaccurate, together with available data about the initial values of the state and some probabilistic description of the system and measurement noises [26].

2) The Kalman Filter data model: The Kalman Filter assumes that the state $\mathbf{x}_k$ can be written in terms of the previous state $\mathbf{x}_{k-1}$, with the linear equation

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1}.$$

The matrix $\mathbf{F}_{k-1}$ relates the previous state with the current state, and $\mathbf{v}_{k-1}$ is the state noise. At time k, the measurement $\mathbf{z}_k$ of the state $\mathbf{x}_k$ can be written as

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k,$$

where $\mathbf{H}_k$ relates the state with the measurement, and $\mathbf{w}_k$ is the measurement noise [9]. The Kalman Filter can then divided into five steps: (1) state estimate extrapolation, (2) error covariance extrapolation, (3) Kalman gain, (4) state estimate update and (5) error covariance update [24].

3) A Basic Model Example: To give the basic idea of how the Kalman Filter works, we use an example from R. Negenborn's master thesis "Robot localization and Kalman filters - on finding your position in a noisy world". The scenario is given as a robot navigating in an environment where it wants to localize itself. We assume that the robot has access to all measurements, and that the robot is subject to noise when it navigates its way around the environment.

First, we create a model over the robot's navigation, i.e the behavior of the system tells us how the location of the robot changes over time. The system that describes the true location x_k of the robot is

$$x_k = x_{k-1} + s + \omega_k,$$

where x_k depends on the previous location x_{k-1}, the speed s (assumed to be constant) and a special noise term ω_k. We assume that the noise is a zero-mean, Gaussian distributed noise, which means that the average value of the noise is equal to zero. The deviation of the noise is denoted by σ_ω. In order to use absolute measurements when estimating the location, we need a description of how the measurements are related to the true location x_k. To do this, we assume that we have a measurement model that can describe the relationship between z_k and x_k as

$$z_k = x_k + v_k,$$

where v_k is the noise that corrupts the measurements. Just as with ω_k, we assumed that v_k is zero on average, Gaussian distributed and has a deviation of σ_v.

Now assume that we have the initial estimated state of the robot's location, $\hat{x}_0$, and an uncertainty (variance) of σ_0^2. To make a prediction of the robot's new position, we assume that the robot drives for one time step. Using the system model, we know that the robot's location will (on average) change s, and now we can update the estimate of the location with this knowledge. We calculate the new location $\hat{x}_1$ at step $k = 1$ as

$$\hat{x}_1 = \hat{x}_0 + s.$$

In this calculation, we took the noise equal to zero, since we do not know how much the current state is corrupted by noise, although we know that it most certainly is. Additionally, we know that the noise varies around zero which also justifies ω_k being set to zero. We would now like to update the uncertainty in our new position, and this we do by calculating σ_1^2 that we have in our new estimate as

$$\sigma_1^2 = \sigma_0^2 + \sigma_\omega^2. \tag{4}$$

If the robot would continue moving without receiving any absolute measurements, the uncertainty in equation (4) would increase. What we then need is an absolute measurement which we use to correct the prediction that we made. If we now assumed that we make an absolute measurement z_1, then we want to use this measurement in our estimation of the location. We do this by including z_1 in a weighted average between the uncertainty in the observed location from the measurement z_1 and the uncertainty in the estimate that we already had in $\hat{x}_1$

$$\hat{x}_1^+ = \frac{\sigma_v^2}{\sigma_1^2 + \sigma_v^2}\hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2} z_1 = \hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}(z_1 - \hat{x}_1). \tag{5}$$

This has two direct consequences: If the uncertainty σ_1^2 in the old estimate is large, then we will include much of the new measurement in the estimation. However, if the uncertainty in the measurement, σ_v^2 is large, then we do not include much of the new measurement.

The absolute measurements provide independent location information and do not depend on earlier location estimates, hence they reduce the uncertainty in the location estimate. We can combine the uncertainty in the old location estimate with the new measurement, which gives us the uncertainty $\sigma_1^{2,+}$ in the new estimate as

$$\frac{1}{\sigma_1^{2,+}} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_v^2},$$

which we can rewrite into

$$\sigma_1^{2,+} = \sigma_1^2 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}\sigma_1^2. \tag{6}$$

We can see in equation (6) that when we add new information, the uncertainty decreases in the final estimation. We now have that $\sigma_1^{2,+}$ is actually smaller or equal to both the old location estimate uncertainty σ_1^2 and the measurement uncertainty σ_v^2. If we look closer at equation (5) and (6), we can see that we have used the same gain K as

$$K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}. \tag{7}$$

Using K, we rewrite (5) and (6) into

$$\hat{x}_1^+ = \hat{x}_1 + K(z_1 - \hat{x}_1),$$

$$\sigma_1^{2,+} = \sigma_1^2 - K\sigma_1^2 = (1 - K)\sigma_1^2.$$

The factor K is now the key essence of the Kalman Filter. It determines how much of the information from a specific measurement should be used when updating the state estimate. This then leads to two extreme cases. If the uncertainty in the last location estimate, σ_1^2, is close to zero, then K will be close to zero. This means that the last received measurement is will not be taken into great account in the state update. If then the measurement uncertainty, σ_v^2, is small then K will approach one. This suggests to that the measurement will be used in the state update [25].

4) Fusing information from multiple cameras: We have showed how a Kalman Filter can be used to estimate the position of a robot. Now, we will show how the Kalman Filter theory can be used to estimate the position of a moving object, when the information of the position is gathered from multiple cameras. Let $\mathbf{x}_k = (x_k, y_k, z_k)^T$ be the position of the object that we want to track at time k. The information gathered from each of the N cameras can be denoted as $\mathbf{u}_k^i = (u_k^i, v_k^i)^T$, $0 \leq i < N$. The projection of the position of the moving object on each image plane can be written as

$$\mathbf{u}_k^i = \mathbf{P}^i \mathbf{x}_k + \mathbf{E}_k^i,$$

where $\mathbf{P}^i$ is the projection matrix for each camera, and $\mathbf{E}_k^i$ is the noise [24].

When tracking a moving object in 3D from its projections on the image planes of the cameras, the standard Kalman Filter cannot be used. The reason is that the relationship between 3D location $\mathbf{x}_k = (x_k, y_k, z_k)^T$, and the projections onto the image planes $\mathbf{u}_k^i = (u_k^i, v_k^i)^T$, are non-linear. The problem can be solved by using the Extended Kalman Filter [24].

5) Extended Kalman Filter: In the Extended Kalman, Filter the process is written as

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1},$$

and the measurement

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{w}_k.$$

Here, $\mathbf{f}_{k-1}$ and $\mathbf{h}_k$, does not have to be linear functions. The Extended Kalman Filter linearize $\mathbf{f}_{k-1}$ and $\mathbf{h}_k$, around the current state for each time step k [9].

For the Extended Kalman Filter to work, one need to select a prediction model, that describes the motion of moving object. To be able to track in real-time, these models have to be efficient, considering the limited bandwidth and computational power of a wireless camera network. For slow moving object with simple motions, a model that only contains information of the location is enough. More complex models demands knowledge of the motion of the object. This is problematic when tracking, e.g., people, because their movement is often unpredictable. Creating a prediction model of, e.g., an AGV on the other hand is achievable, due to the fact that the AGV could have a predetermined route to follow. Therefore, using a model of the motion is evident when information of the motion is available in advance. [2].

D. Localization of an AGV

Project D4 works with navigating an AGV (Autonomous Ground Vehicle) indoors. The problem is that they need to know their position to be able to navigate through rooms. Figure 10 shows an example of the map that the AGV uses to navigate. The red shapes are obstacles which the AGV should avoid, and the dotted path is the path that the AGV should follow. By using a wireless camera network, the AGV could be detected and tracked. By having communication with the AGV through a base station, the camera network could send the position to the AGV. Using the obtained the position, the AGV could localize itself on the map in Figure 10.

D4 reported that obtaining position information at a frequency of about 2,5 Hz would be good. When implementing our tracking algorithm in MATLAB, we achieved that we could send coordinates with a frequency of 2 Hz. This however was performed with a resolution of 1280×1024 pixels. Such high resolution is not necessary when detecting and tracking an AGV. By lowering the resolution, the frequency could be increased several times and satisfy the required frequency. Also, by using multiple cameras combined with a Extended Kalman Filter would make the position information even more precise.

The AGV have a UWB sensor, which could scan the room for objects, so that it could send information to the base station about certain objects that maybe are invisible to the static cameras in the room. Such fusion with our tracking algorithm and the AGV sending important information about the situation, could be very useful for example in a "search and rescue mission".

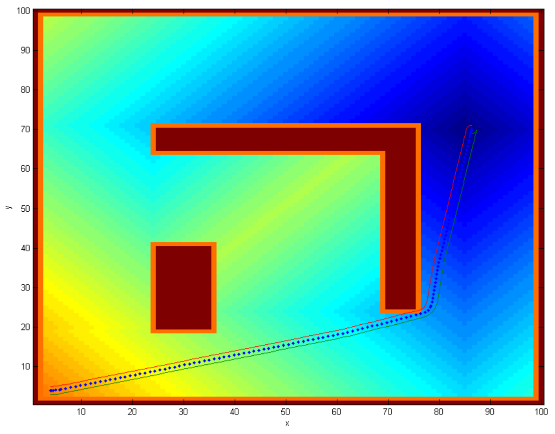

Fig. 10. An example of the map that the AGV uses to navigate

VI. New Application - The Smart Illumination system

A. Existing Applications

Before presenting our own new application of wireless camera networks, we will start with a survey of existing interesting applications that we have found. The most common applications of wireless camera networks that we have encountered is for surveillance. Instead of having a human detecting intruders, the camera itself can detect, trigger an alarm, and then give the position of the intruder. In surveillance, the wireless camera network can also be used to detect and study suspicious behavior of people, e.g., if it looks like someone is stealing something.

Other interesting applications that we have found is for health care. In the master thesis "Multi-Camera Person Tracking using Particle Filters" the authors suggests an application for monitoring elderly in their homes. The application uses a wireless camera network to detect abnormal behavior and accidents, e.g., if a person has fallen and lies on the floor. This is done by studying the body posture of the person. The purpose of this application is to let elderly live a safe life in their own homes for a longer time and with minimal assistance [9].

We have also found applications that create intelligent environments. In the report "Multi-Camera Multi-Person Tracking for Easyliving" the authors present a system that can automatically start and stop a DVD when a person sits or stands up from a couch. The system can also play the DVD on different displays in the room, depending on where the person currently is [27].

B. The Smart Illumination system

For our own application we have considered a topic of current interest: energy consumption and environmental awareness in a smart building. Our idea is to combine simple motion detectors and low power wireless camera networks in order to help the user save both energy and money, without making everyday life less comfortable.

According to the Swedish Energy Agency, lighting consumes around 20% of the total energy consumption in domestic homes, which roughly adds up to 4000 SEK/year in energy costs [28]. Nowadays there are several campaigns trying to make the public aware of its energy consumption and give us solutions on how to save both energy and money. So far, they all involve more or less a sacrifice from the user, e.g., we are told not to raise the heaters in our homes too much and instead wear more clothes inside, or that we should constantly be aware of all the lights we have in our home and actively turn them on or off when leaving or entering a room. Although being good advice in terms of energy consumption, they all comprise our comfort and it is probably one of the main reason why people do not give in into environmentalists' ideas and solution.

With our application, "The Smart Illumination System", we want to solve the problem with idle lighting in homes in order to save energy, and in the same time make everyday life more comfortable. As a part of the smart building, our application will help the user being more environmental friendly and save money without basically doing anything. We know that we should turn off the lights when we are leaving a room or if we are not directly using it. However, turning on and off lights is a very impractical and time consuming thing to do, which in most cases leads to the lighting being kept on. Perhaps sometimes we even forget that we left the lights on in a room and it could be left on for hours, draining energy. There is a simple solution to this problem existing today that is using motion detectors to detect when a person is entering a room, and then activating the light. When there has not been any motion for a while the system turns off the light. Our application is a further development of this concept.

The basic principle of what we want to do, with the help of motion detectors and wireless camera networks, is to optimize the illumination of a room. When a person is entering the room, motion detectors will turn the light on. Then the wireless camera network will track the position of the person and then adjust the light, depending on where the person is and what the person is doing. Instead of a very simple system which can only turn all lights on or off, our smart system can also dim and individually control each light source in the home. Each camera is aware of its position and the layout and structure of the room it monitors. Using object detection and tracking, the camera network can determine the position of a person, and with a pre-programmed map of the room, the wireless camera network will be able to tell if a person is, e.g., sitting by the desk reading or watching a movie in the sofa. These special sections in the room are represented as zones in the pre-programmed map. Depending on the user's preference, the lighting for each zone can be set individually. If the user would sit down by his desk, then his desktop light will automatically be turned on. If the user would then move to the TV couch, the surrounding lights would dim down in order to give the optimal TV lighting.

The system will of course be able to shut down the lights whenever the room is empty, and depending on which direction the person left the room, the adjacent room will light up in advance. This will prevent the lights from being turned on and off repeatedly, avoiding the effect of blinking lights, which is a problem when using only motion detectors. The system would also have a very intelligent feature that we have called activity recognition, which will be able to combine the position with what the person is doing. For example, if a person would sit in the sofa reading a book, the system will adjust the light for reading. However, if the person starts watching TV, still sitting in the same sofa, the light will change and become optimized for TV watching. Using information from camera networks to recognize specific behaviors, such as reading, is usually far more complicated than just finding the position of a person. Therefore, this feature of our application would need more work to develop than the previously mentioned.

With the help of smart object detection and tracking, the Smart Illumination System would work great in everyday life with minimal human interacting. When the system is installed and running, the user will hopefully use it without even thinking of it, and at the same time save energy. A question that might have occurred is about the energy consumption of the cameras, compared with letting the lights be turned on. The camera node CMUcam3 that was mentioned earlier, consumes 0,65 W, which is considerably lower than a standard lightbulb that consumes 60 W. A corresponding low energy lightbulb use 15 W, which is still higher than the camera node. The conclusion is that if the Smart Illumination System can turn off just one, at the moment unnecessary light, there will be energy savings.

One drawback of our application, is the work that has to be done installing the system. For example, the zones and which light that should be active when the user is present, has to be determined and programmed into the system.

Optimally, the system would be installed in new buildings, where cameras can be mounted in places where it does not bother the user, and the zones can be determined in advance. The system would be useless and very irritating if it did not work properly. Just imagine having lights automatically turned off when needing them.

Our system needs a central computer, which is connected to the electronic devices in the home, i.e., the wireless cameras and lights, in order to operate. This central computer or “brain”, does not have to be big and could be hidden somewhere in the house. In older houses, it is not common to have a central computer installed that controls the house. Lately, there has been a lot of progress in the field and major energy companies are today discussing the possibility to have a small computer in the home which controls, e.g., energy consumption or ventilation, etc. In the smart illumination system, the wireless cameras will send the position information to this central computer, which by knowing the zones can determine which light to turn on or off. The aim is that the computer should have an easy-to-use user interface, so that the user can easily reprogram the zones of the a room, in the case of, e.g., a refurnishing.

The Smart Illumination System is ahead of its competition, which is normal motion detectors. It might be that our application is too advanced and too hard to incorporate in old buildings. However, in new building productions, where energy efficiency is an important factor, the smart illumination system would be easier to install and incorporate with the rest of the building’s technology. A wireless camera network in a smart building, can be used for many other application besides optimizing lighting. For example, it can be incorporate with other building functions, such as ventilation, heating, and the usage of domestic robots.

VII. Discussion

This project has been interesting to work with, because it has involved many fields, such as image processing and automation control. The problem formulation we started with was wide, with many different issues to address. We have been able to go deeper and implement some parts of the project, while other parts off the project would take much more time to completely address. For example, more knowledge and work from our side, would be necessary to fully understand how a Extended Kalman Filter could be implemented in a wireless camera network. On the other hand, problems like how to perform background subtraction, have been well described and demonstrated. Overall, we have gained a better understanding for problems and possibilities concerning moving object detection and tracking, using wireless camera networks and UWB sensors.

If we start with discussing the results of our own implementations, we found that our algorithms for background subtraction worked well. With the right choice of threshold value and a distinct background, the algorithms could accurately separate the foreground from the background. The differences between offline background subtraction and frame-by-frame subtraction was evident during our testing of the algorithms. With the noise filter we were able to raise the threshold, and in that way avoid missing important information about objects and at the same time reduce noise. We found that the filter could be set to about 90-99%, without losing any important parts of the moving object. Although being good results, it is important to mention that the testing of the algorithms was performed with a distinct background, not containing details or strong backlight. This scenario represents the ideal environment for the algorithms, however, it is not very realistic. Changes in lighting were evidently a problem, also if the object color matched the wall too much, the algorithms would fail to detect that object.

Our tracking algorithm worked well and was efficient. However, the position information was a not exact, due to that algorithm calculated the position from the center of a rectangle who framed the object, and did not consider the real shape. A further development would, instead determine the contour of the object, calculate the mass centrum, and then return that as the position. We where able to run the algorithm in real-time with a frame rate of 2 fps, with a resolution of 1280 × 1024 on the camera. A lower resolution would increase the fps significantly. Our tracking algorithm that could handle multiple object, did also have a frame rate close to 2 fps, but to make it precise showed to be difficult. The multiple object tracking algorithm had two evident flaws: it would sometimes fail to capture the whole object and its borders, and sometimes it would make two objects seem as one if they were close to each other.

Future work would be to produce a more intelligent algorithm for multiple object tracking. It would also be interesting to try implementing more mathematically advance methods for moving object tracking, such as optical flow and compare it with the background subtraction algorithms. We would also like to implement an adaptive background subtraction algorithm, that could tackle some of the difficulties we found with our implemented algorithms, such as lighting change and varying background.

Another interesting aspect that could be implemented, is an algorithm for object recognition. For example, a algorithm that can determine whether the moving object is a human or AGV, etc. Also an algorithm for behavioral recognition, which we planned to use in “The Smart Illumination System”, could be implemented. We see many applications for behavioral recognition, especially in health care, detecting when people get ill, or in other emergencies.

The next step would be to create our own wireless camera network, where we could try to fuse multiple cameras. During this project we only implemented algorithms for one single camera, and it would have been interesting to implement tracking algorithms that could use information from multiple cameras. We have worked with methods on how to perform 3D localization using multiple cameras. However, the leap from what we did accomplish, to in reality perform 3D localization in a wireless camera network was too big. It would also be interesting to implement a Kalman Filter, to gain a better understanding of how it works.

Even though the technology for a efficient wireless camera network does exist, we still see potential for further development. For example, if the cameras could see in the dark, the wireless camera network could be used during more hours of the day. Our supervisor has told us that such a camera is in development. In the smart building, energy efficiency is an important factor. Therefore, development smart cameras with low energy consumption, but still with a high computational power would appeal.

A major issue with realizing this project is the topic of personal integrity and ethics. Although the intention of camera networks is harmless, the feeling of being constant monitored is a major concern for people today. We sometimes hear about protests against increased number of surveillance cameras in society, so why would there be any less concern about installing cameras in our homes? There have been incidents where live video streams from installed camera networks in retirement homes, accidentally have been available online for public access. Despite these incidents being rare, the question still remains whether or not someone is watching you through your own personal camera network. A possible solution to ensure the video data being secure is to have the camera network working offline, i.e. no Internet connection to the central processing unit or the cameras. To give the network limited storage memory could also force the network to never save any data, and instead only process the information once and then discard it. One could also avoid having cameras in private rooms, such as personal bedrooms and bathrooms. However, if people feel that a majority of their rooms are not suitable for surveillance, then why would they want to install the system in the first place? These camera networks are supposed to be a part of the smart building, but if we only install the camera networks in a few rooms, we would end up with several independent smart rooms, and the whole idea with an intelligent home would fall apart.

For the wireless camera network to become a part of the future smart building, we need to make people understand the benefits of a camera network. Once people have realized the possibilities, with the applications of a wireless camera network, we think that it will become a part of every home. As with almost all the new technologies, it will take a while for people to understand what the purpose is, and what it can be used for. What we arrive at after this project, is that we see potential for wireless UWB and camera networks to be incorporated in the smart building.

VIII. Acknowledgments

We would like to thank our teaching assistant Farhad Farokhi, for his useful advices and support through the project. We would also thank Erik Ehrlin and Martin Törnqvist in D4, for a smooth collaboration. Finally we would like to thank our supervisor Carlo Fischione, for giving us an interesting project and feedback on our work.

References

[1] Wikipedia.org. Wireless sensor network. [Online]. Available: http://en.wikipedia.org/wiki/Wsn

[2] J. Sanchez-Matamoros, J.-d. Dios, and A. Ollero, "Cooperative localization and tracking with a camera-based wsn," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, april 2009, pp. 1 –6.

[3] I. Singh, "Real-time object tracking with wireless sensor networks," Master's thesis, Lulea University of Technology, Sweden, 2007.

[4] M. van de Goor, "Indoor localization in wireless sensor networks," Master's thesis, Radboud Universiteit Nijmegen, the Netherlands, 2009.

[5] Wikipedia.org. Visual sensor network. [Online]. Available: http://en.wikipedia.org/wiki/Visual_sensor_network

[6] S. Stanislava and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, p. 21, 2009.

[7] M. Akdere, U. Cetintemel, D. Crispell, J. Jannotti, J. Mao, and G. Taubin, "Data-centric visual sensor networks for 3d sensing," in *GeoSensor Networks*, ser. Lecture Notes in Computer Science, S. Nittel, A. Labrinidis, and A. Stefanidis, Eds. Springer Berlin / Heidelberg, 2008, vol. 4540, pp. 131–150.

[8] B. Tavli, K. Bicakci, R. Zilan, and J. Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools and Applications*, pp. 1–38, 2011.

[9] M. Andersen and R. Skovgaard-Andersen, "Multi-camera person tracking using particle filters - based on foreground estimation and feature points," Master's thesis, Aalborg University - Department of Electronic Systems, Denmark, 2010.

[10] A. Sankaranarayanan, A. Veeraraghavan, and R. Chellappa, "Object detection, tracking and recognition for multiple smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1606 –1624, oct. 2008.

[11] Y. Dedeoglu, "Moving object detection, tracking and classification for smart video surveillance," Master's thesis, Bilkent University - The Institute of Engineering and Science, Turkey, 2004.

[12] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456 –1477, oct 2001.

[13] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, 2006.

[14] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 3, pp. 334 –352, aug. 2004.

[15] B. Bhanu, C. V. Ravishankar, A. K. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, *Distributed Video Sensor Networks*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[16] Wikipedia.org. Ultra wide-band. [Online]. Available: http://en.wikipedia.org/wiki/Ultra-wideband

[17] O. Hirsch, M. Janson, W. Wiesbeck, and R. Thoma and, "Indirect localization and imaging of objects in an uwb sensor network," *Instrumentation and Measurement, IEEE Transactions on*, vol. 59, no. 11, pp. 2949 –2957, nov. 2010.

[18] R. Thoma, O. Hirsch, J. Sachs, and R. Zetik, "Uwb sensor networks for position location and imaging of objects and environments," in *Antennas and Propagation, 2007. EuCAP 2007. The Second European Conference on*, nov. 2007, pp. 1 –9.

[19] A. Yarovoy, L. Ligthart, J. Matuzas, and B. Levitas, "Uwb radar for human being detection," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 21, no. 3, pp. 10 – 14, march 2006.

[20] M. Arik and O. Akan, "Collaborative mobile target imaging in uwb wireless radar sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 6, pp. 950 –961, aug. 2010.

[21] B. Harker, A. Chadwick, and G. Harris, "Ultra-wideband 3-dimensional imaging (uwb 3d imaging)," *Roke Manor Research Limited*, 2008.

[22] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," in *Proceedings of the 13th annual ACM international conference on Multimedia*, ser. MULTIMEDIA '05. New York, NY, USA: ACM, 2005, pp. 229–238.

[23] C. Seres, "Robot navigation in indoor environment," Master's thesis, University of Southern Denmark, Denmark, 2008.

[24] C. Canton-Ferrer, J. R. Casas, A. M. Tekalp, and M. Pardàs, “Projective kalman filter: multiocular tracking of 3d locations towards scene understanding,” in *Proceedings of the Second international conference on Machine Learning for Multimodal Interaction*, ser. MLMI’05. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 250–261.

[25] R. Negenborn, “Robot localization and kalman filters - on finding your position in a noisy world,” Master’s thesis, Utrech University, The Netherlands, 2003.

[26] P. Yves, “Dynamic objects tracker in 3d,” Master’s thesis, EidgenŽssische Technische Hochschule, Switzerland, 2011.

[27] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, “Multi-camera multi-person tracking for easyliving,” in *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, 2000, pp. 3 –10.

[28] Energimyndigheten. (2011, nov) Belysning. [Online]. Available: http://energimyndigheten.se/Hushall/Din-ovriga-energianvandning-i-hemmet/Hembelysning/